REMARKS

Claims 1-67 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Provisional Double Patenting Rejection:

The Office Action rejected claims 1, 21, 38, 53, 66 and 67 under the judiciary created doctrine of obviousness-type double patenting as being unpatentable over claims 1, 18, 36, 43, 50, 57, 61, 64 and 65 of co-pending Application No. 10/626,420. Given that this rejection is provisional, Applicants will consider filing a terminal disclaimer or arguments traversing the obviousness-type double patenting rejection once the rejection becomes non-provisional or the present application is otherwise in condition for allowance.

Section 103(a) Rejections:

The Office Action rejected claims 1, 4-10, 19, 21-26, 38-42, 48, 52-60, 62, 66 and 67 under 35 U.S.C. § 103(a) as being unpatentable over Takahashi et al. (U.S. Publication 2002/0194237) (hereinafter "Takahashi") in view of Chen et al. (U.S. Publication 2002/0116430) (hereinafter "Chen") and further in view of Yeh et al. (U.S. Patent 6,848,043) (hereinafter "Yeh"), claims 2, 3, 15-18, 27-29, 35 and 43-46 as being unpatentable over Takahashi in view of Chen and further in view of Yeh and Vanstone et al. (U.S. Patent 6,735,611) (hereinafter "Vanstone"), claims 11, 20, 30, 31, 37, 47 and 61 as being unpatentable over Takahashi in view of Chen and further in view of Yeh and Stribaek et al. (U.S. Patent 7,181,484) (hereinafter "Stribaek"), and claims 12-14, 32-34, 49-51 and 63-65 as being unpatentable over Takahashi in view of Chen and further in view of Yeh and Chen et al. (U.S. Patent 6,687,725) (hereinafter "Chen2"). Applicants respectfully traverse these rejections for at least the following reasons.

Regarding claim 1, the cited references fail to teach or suggest *a first arithmetic circuit comprising a first plurality of arithmetic structures feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set in the public-key cryptography application, generated by the first arithmetic circuit, to a second arithmetic circuit comprising a second plurality of arithmetic structures*. In rejecting claim 1, the Examiner cites paragraphs [0009] and [0017] of Takahashi as disclosing these aspects of the claimed invention. As noted in paragraph [0009], Takahashi is directed to a multi-function processor that can perform multiple types of modulo mathematic operations (e.g., modulo multiplication, modulo exponentiation, and modulo reduction).

As noted in Applicants' previous response, paragraph [0017] of Takahashi describes that one or more of such processors can be included in a system for encrypting/decrypting data. This paragraph also describes that each such processor receives operands for a particular modulo mathematic operation, stores the operands in an operand storage portion, performs the particular modulo mathematic operation using the stored operands, and outputs a final result after iteratively computing a running partial product and post-processing the final partial product. Other portions of Takahashi describe that different bits of the originally received and stored operands for the particular modulo mathematic operation are supplied to different pipeline stages of the processor for computing the running partial product. In other words, Takahashi describes a processor that is configured to perform one modulo mathematic operation at a time, using operands that are explicitly supplied to it for that operation. For example, each of paragraphs [0010] – [0016] describes a different modulo mathematic operation (e.g., a modulo multiplication type operation, such as AB mod N, or a modulo N exponentiation, such as A^E mod N) that can be performed by the multi-function modulo processor of Takahashi. As described in these and other paragraphs in Takahashi, for each of these mathematic operations, all of the operands for the operation are explicitly provided to (received by) the processor for that operation and are used in computing a result. Contrary to the Examiner's assertion, there is nothing in the cited passages, or elsewhere in Takahashi, that describes feeding back high order bits of a previously executed single

arithmetic instruction of a processor instruction set that were generated by a first arithmetic circuit, to a second arithmetic circuit. Furthermore, Takahashi does not describe any instructions of the processor's instruction set, much less any type of feedback that occurs between two different instructions (i.e. two instances of any of the instructions of the processor's instruction set) when they are executed.

Further regarding claim 1, the cited references fail to teach or suggest *the second arithmetic circuit generating a first partial result of a currently executing single arithmetic instruction of the processor instruction set in the public-key cryptography application, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result representing the high order bits summed with low order bits of a result of a first number multiplied by a second number, the summing of the high order bits being performed during multiplication of the first number and the second number, the summing and at least a portion of the multiplication being performed in the second arithmetic circuit*. The Examiner cites paragraphs [0017], [0038], and [0044] of Takahashi as disclosing these aspects of Applicants' invention. The Examiner notes that these paragraphs describe that the pipeline portion of the processor receives operands from the operand storage portion of the processor. As discussed in Applicants' previous response, and contrary to the Examiner's suggestion, the operands that are stored in the operand storage portion of the processor and that are received by the pipeline portion of the processor are not high order bits of a previously executed single arithmetic instruction of the processor's instruction set. No such feedback from a previously executed instruction to another (currently executing) instruction is disclosed in Takahashi. Instead, the operands that are stored in the operand storage portion of the processor and that are received by the pipeline portion of the processor are operands that were received by the processor for the currently executing modulo mathematic operation.

The Examiner submits that paragraph [0017] of Takahashi describes, "pipeline storage processing (multiplication, summation) stage iteratively computing a running partial product using one or more received operands a predetermined number of times;

post processing stage to receive final partial product and compute result" (emphasis added). **Applicants note that in these remarks, the Examiner himself admits that the pipeline is computing a running partial product using one or more operands that were received for the currently executing modulo mathematic operation**. As in the previous Response, Applicants again assert that there is nothing in Takahashi describing that any of the received operands are high order bits fed back from a previously executed single arithmetic instruction. The Examiner submits that paragraph [0044] discloses, "receive bit stored in the current highest order position; first carry-save processor (a0 position); second carry-save processor (a1 position); third carry-save processor (a2);…)" The Examiner's suggestion that this passage describes an arithmetic circuit of Takahashi receiving high order bits fed back from a previously executed single arithmetic instruction are completely unsupported in the reference itself. The cited passage *actually* states, in its entirety (with emphasis added):

> [0044] Each of the carry save processors 422-1, 422-2, 422-3, 422-4 is coupled to receive a single bit of data stored in the first operand register 414, all of the data bits stored in the second operand register 416, and all of the data bits stored in the third operand register 418. Specifically, with respect to the data stored in the first operand register 414, the first carry-save processor 422-1 is coupled to receive the single data bit stored in the least significant bit position of the first operand register (e.g., the a.sub.0 position), the second carry-save adder 422-2 is coupled to receive the single data bit stored in the next position of the first operand register 414 (e.g., the a, position), third carry-save adder 422-3 the next (e.g., the a.sub.2 position), and the fourth carry-save adder 422-4 the next (e.g., the a.sub.3 position). As will become more apparent when a discussion of the iterative calculations performed by the pipeline processing unit 306 is provided, the data stored in the first operand register 414 is shifted to the right four bits after each iteration, until all of the data bits stored in the first operand register 414 are utilized in the calculational process.

In other words, this passage describes, in detail, how the individual bits of the operands that were received (and stored) for a single modulo mathematic operation are distributed to different carry-save adders in order to perform the desired calculation. **This teaches absolutely nothing about the above-referenced limitations of Applicants' claim**. For example, nothing in the cited passage, or elsewhere in

Takahashi, discloses a (second) arithmetic circuit that generates a first partial result representing the high order bits (i.e. high order bits fed back from a previously executed single arithmetic instruction and not explicitly specified as a source operand for a currently executing instruction) summed with low order bits of a result of a first number multiplied by a second number, the summing of the high order bits being performed during multiplication of the first number and the second number, the summing and at least a portion of the multiplication being performed in the second arithmetic circuit, as in Applicants' claim.

Applicants' claim describes a relationship between a currently executing single arithmetic instruction of the processor instruction set and a previously executed single arithmetic instruction of the processor instruction set. Thus, the claim defines a relationship between the executions of two successive single arithmetic instructions of the processor instruction set. Applicants assert that the Examiner has not identified a single arithmetic instruction of a processor instruction set (i.e. an instruction implemented in a processor) in Takahashi whose execution causes the performance of the specific collection of operations described in Applicants' claim by the circuitry disclosed in Takahashi, or results in the relationships between successive instruction executions (i.e. between the executions of two distinct instances of single arithmetic instructions) recited in the claim.

Takahashi describes operations of a modulo processor architecture that are used to implement a single arithmetic instruction (one of several modulo mathematic operations that can be performed by the processor). Takahashi describes feedback between operations of the processor that collectively implement this single arithmetic instruction. Takahashi does not describe implicit feedback (or an implicit relationship) between this single arithmetic instruction and a previously executed single arithmetic instruction of the processor's instruction set, as required by Applicants' claim, i.e. adding high order bits of a previously executed single arithmetic instruction without specifying these bits as a source operand of the currently executing single arithmetic instruction. Applicants assert that Takahashi clearly does not disclose these aspects of Applicants' claimed invention.

In the Response to Arguments section of the Office Action mailed September 27, 2011, the Examiner repeatedly states, "Takahashi discloses a processor capability of performing multiple pipeline operations from a single instruction." Applicants again assert that in Takahashi, the "multiple pipeline operations" referenced by the Examiner are performed not merely from (or in response to) a single instruction, but are performed exclusively for a single instruction (i.e. in order to complete the execution of the single instruction in the pipeline). Applicants further assert that there is no relationship between these pipeline operations and any operations performed in the pipeline for a different single instruction, much less the implicit feedback of a partial result of a previously executed single instruction for use in a currently executing single instruction, as in Applicants' claim.

On pages 14-15 of the Office Action mailed September 27, 2011, the Examiner admits that Takahashi does not specifically disclose a partial result that comprises a high order portion of a result and relies on Chen to teach this aspect of Applicants' claim. Specifically, the Examiner cites paragraph [0017], lines 14-19 as disclosing "hardware utilized to perform addition and multiplication"; paragraph [0100], lines 12-18 as disclosing, "output is high order bits, storing high order bits from adder"; and paragraph [0274], lines 16-19 as disclosing, "carry feedback out of high order position." Chen is directed to a standalone engine in which a modular exponentiation function is implemented. In Chen, to perform multiplication and addition operations on large arrays, the arrays are partitioned into smaller structures that are operated on by a plurality of chained processing elements in a two-phased pipelined fashion. In other words, like Takahashi, Chen is directed to methods for performing a single modular multiplication operation in a pipelined fashion.

The cited portions of Chen describe the operations of the different processing elements (which share some hardware) in different phases of the execution of the single modular multiplication operation. For example, paragraph [0274] describes that each processing element chain outputs results of a single modular multiplication operation 2k

bits at a time, describes how these results are combined, and describes generating checksum values and/or error signals for the overall result of the single modular multiplication operation. The multiple adders utilized in executing the single modular multiplication operation in Chen's system all perform modular addition and include a carry feedback out of the high order position into the low order position. However, this teaches nothing about a partial result from a previously executed single arithmetic instruction that is not explicitly specified as a source operand for a currently executing single arithmetic instruction being implicitly added to a result of a first number multiplied by a second number, as in Applicants' claim, whether that partial result is the high order bits or any other portion of the result of a previously executed single arithmetic instruction. No such relationship between the executions of different single arithmetic instructions is taught or suggested by the cited references, whether they are considered alone or in combination.

On page 15 of the Office Action mailed September 27, 2011, the Examiner submits, "It would have been obvious to one of ordinary skill in the art to modify Takahashi for a partial result comprises a high order portion of a result as taught by Chen. One of ordinary skill in the art would have been motivated to employ the teachings of Chen for the benefits achieved from the efficiency provided by a partitioning mechanism that enables the sharing of hardware. (Chen paragraph [007], lines 11-13)." However, the cited portion of Chen describes a benefit of partitioning a single modular multiplication operation into different operational phases, not a benefit of adding a partial result from a previously executed single arithmetic instruction that is not explicitly specified as a source operand for a currently executing single arithmetic instruction being to a result of a first number multiplied by a second number, or a benefit of that partial result being a high order portion of the result of the previously executed instruction, which Chen does not disclose. Therefore, the Examiner's stated reason to combine the references is not supported in the references themselves. In addition, since neither reference teaches or suggests implicitly adding a partial result from a previously executed single arithmetic instruction that is not explicitly specified as a source operand for a currently executing single arithmetic instruction being to a result of a first number multiplied by a second

number, the combination suggested by the Examiner would not result in Applicants' claimed invention.

On page 15 of the Office Action mailed September 27, 2011, the Examiner further admits that Takahashi-Chen does not specifically disclose feeding back a result (a number of bits) of a previously executed single arithmetic instruction and relies on Yeh to teach this aspect of Applicants' invention. Specifically, the Examiner states, "Yeh discloses feeding back bits of previously executed single arithmetic instruction (an operand transfer between two single instructions). (*see* Yeh, col. 5, ll. 27-33; first instruction in a dependency chain accepts input from registers; second instruction accepts results of first instruction as it input operand)." Yeh is directed to methods and apparatus for improving system performance when using redundant arithmetic to execute instructions in a dependency chain. For example, in Yeh, if the result of one instruction performed using redundant arithmetic is specified as an input to another instruction that is to be performed using redundant arithmetic, the apparatus may use the stored result of the first instruction (which was output in redundant form) as the input to the second instruction without having to convert it to redundant form. The instructions in the dependency chain may execute separately from instructions not in the dependency chain.

The cited portion of Yeh states, "The first instruction in a dependency chain operates with its input operands in the conventional form. For example, the first instruction in the dependency chain may accept its input operands from the registers in the conventional form. The result of the first instruction is in the redundant form. The second instruction in the same dependency chain then accepts the result of the first instruction as its input operand." Applicants note that (in contrast to the above-referenced limitations of Applicants' claim) the inputs (operands) of the second instruction are explicitly specified for that instruction and do not change. In order for Yeh's apparatus to take advantage of a dependency chain to improve performance, the dependency chain must already exist due to the explicit identification of the outputs and input operands in the instructions in the dependency chain. Only the source of the redundant form of an explicitly specified input operand for a second instruction in a

dependency chain is affected by the techniques disclosed by Yeh. In other words, Yeh clearly does not disclose that a result (or a partial result, whether high order bits or not) is fed back from a previously executed single arithmetic instruction and not explicitly specified as a source operand for a currently executing instruction, as in Applicants' claim. Therefore, combining the teachings of Yeh with Takahashi and Chen would not result in Applicants' claimed invention.

The Examiner further submits, "It would have been obvious to one of ordinary skill in the art to modify Takahashi-Chen for feeding back a result (a number of bits) of a previously executed single arithmetic instruction as taught by Yeh. One of ordinary skill in the art would have been motivated to employ the teachings of Stribaek for the benefits achieved from different approaches to improve the performance of arithmetic operations within computing systems." Applicants note that it appears the Examiner cut and pasted these remarks from a different Action, or a different portion of the instant Action, since the "Stribaek" reference is not cited in the rejection of claim 1. In addition, the Examiner has provided no evidence that the system of Takahashi-Chen would be improved by the referenced teachings of Yah, since Takahashi-Chen does not describe dependency chains of instructions to be performed using redundant arithmetic. Therefore, Applicants assert that the Examiner has not stated a proper reason to combine the references, and that the suggested combination would not result in Applicants' claimed invention.

For at least the reasons stated above, Applicants assert that the Examiner has failed to establish a *prima facie* rejection of claim 1.

Claims 38 and 66 include limitations that are similar to those of claim 1 discussed above, and the Examiner rejected claims 38 and 66 for the same reasons as claim 1. Therefore, the arguments presented above apply with equal force to these claims, as well.

Claims 21, 53, and 67 include limitations that are similar to those of claim 1 discussed above, and the Examiner rejected claims 38 and 66 for the same reasons as claim 1. Therefore, the arguments presented above apply with equal force to these

claims, as well. As noted in Applicants' previous Response, these claims include limitations that are not included in claim 1 and which the Examiner did not address in his rejection of these claims. Therefore, the rejection is improper. For example, claim 21 includes the following: *supplying a third number to the second arithmetic circuit*; and *the second arithmetic circuit generating a first partial result of a currently executing single arithmetic instruction of the processor instruction set in the public-key cryptography application, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result being a representation of the high order bits summed with low order bits of a result of a first number multiplied by a second number and with the third number, the summing being performed during multiplication of the first number and the second number, the summing and at least a portion of the multiplication being performed in the second arithmetic circuit*. Applicants assert that nothing in the cited references teaches or suggests these limitations of claim 21, whether they are considered separately or in combination. Claims 53 and 67 include limitations similar to those of claim 21. Therefore, the arguments presented above and directed to claim 21 apply with equal force to these claims, as well.

For at least the reasons stated above, Applicants assert that the Examiner has failed to establish a *prima facie* rejection of claims 21, 38, 53, 66, and 67.

Applicants assert that numerous ones of the dependent claims recite further distinctions over the cited art. Applicants traverse the rejection of these claims for at least the reasons given above in regard to the claims from which they depend. However, since the rejections have been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time. Applicants reserve the right to present additional arguments.

CONCLUSION

Applicants submit the application is in condition for allowance, and an early notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/6000-31500/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Applicants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: December 21, 2011